На правах рукописи

Камашев Мстислав Андреевич

« МОДЕЛИ ОБЪЕКТНО – ОРИЕНТИРОВАННЫХ СУБД ДЛЯ ИНФОРМАЦИОННО-РАСЧЕТНЫХ ЗАДАЧ»

Специальность: 05.13.11 - математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей.

А в т о р е ф е р а т

диссертации на соискание ученой степени

кандидата физико-математических наук

Казань – 2011

Работа выполнена на кафедре теоретической кибернетики ФГАОУВПО «Казанский (Приволжский) федеральный университет»

Научный руководитель: Кандидат физико-математических наук, доцент ФГАОУВПО «Казанский (Приволжский) федеральный университет» Еникеев Арслан Ильясович

Официальные оппоненты: 1.Доктор физико-математических наук, профессор ФГАОУВПО «Казанский (Приволжский) федеральный университет» Ш.Т. Ишмухаметов

2.Кандидат физико-математических наук, заведующий отделом Института информатики АН РТ И.И.Макаров

Ведущая организация: Московский государственный технический университет имени Н.Э.Баумана

Защита диссертации состоится 20 октября 2011г. в 15.30 на заседании диссертационного совета Д. 212.081.24 при ФГАОУВПО «Казанский (Приволжский) федеральный университет» по адресу: 420008, г. Казань, ул. Кремлевская, Д.18, конференц-зал научной библиотеки им. Н.И.Лобачевского.

С диссертацией можно ознакомиться в научной библиотеке им. Н.И. Лобачевского при ФГАОУВПО «Казанский (Приволжский) федеральный университет». Автореферат диссертации опубликован на сайте ФГАОУВПО «Казанский (Приволжский) федеральный университет» (www.ksu.ru).

Автореферат разослан «16» сентября 2011 г.

Учёный секретарь
диссертационного Совета,
к.ф.-м.н, доцент А.И.Еникеев

Общая характеристика работы

Актуальность темы исследования

Одним из важных аспектов современного подхода к разработке программного обеспечения является использование формализованных средств, с помощью которых создаются модели соответствующих программных систем с целью однозначного описания и исследования свойств создаваемых программных средств. Эффективность такой модели определяется не только наличием адекватных средств описания и исследования, но и главным образом в идеале предполагает создание средств автоматической генерации программных средств из соответствующих формализованных спецификаций. Таким образом конечной целью формализованного подхода является создание многоуровневой интегрированной среды разработки программных средств, включающей в себя формализованные средства для построения модели и средства автоматической генерации программ из их спецификаций. Однако формализм большинства предлагаемых в настоящее время методов, основывается, как правило, на сложных математических средствах высокого уровня и следовательно является труднодоступным на этапе практической реализации моделей. Поэтому задача интеграции традиционных полуформальных методов с формальными представляется весьма актуальной. Опыт соответствующих разработок в этой области показывает, что универсальные подходы к решению указанной выше задачи не дают желаемых результатов и наибольший эффект достигается на пути построения специализированных систем моделирования.

Представляемая работа посвящена построению объектно-ориентированной модели программных систем, ориентированных на автоматизацию решения так называемых информационно – расчетных задач, в число которых включаются задачи компьютерной бухгалтерии, делопроизводства, статистики и т.п. Основной особенностью упомянутого класса задач является использование относительно простых структур данных, в большинстве случаев адекватно представляемых аппаратом реляционной алгебры. Однако, такое важное понятие, как иерархия, определяющее один из основных принципов объектно-ориентированного подхода, в рамках реляционной модели представляется далеко неадекватными способами, что может существенно отразиться на эффективности функционирования соответствующей интегрированной среды разработки программных приложений. Особенностью предлагаемого подхода является расширение реляционной модели путем включения аппарата алгебраических спецификаций и средств функционального программирования с целью гибкого сочетания простоты реляционной модели со средствами эффективного представления иерархически организованных структур.

Цель работы

Основной целью диссертационной работы является методология построения специализированной модели, ориентированной на разработку программных приложений для автоматизации решения информационно-расчетных задач на основе соединения алгебраических спецификаций и средств функционального языка программирования с целью последующей реализации модели в реляционной среде.

Перечень решаемых задач

1. Разработать специализированную модель для класса информационно-расчетных задач на основе расширения реляционной модели средствами алгебраических спецификаций и функционального языка программирования.
2. Создать интегрированную среду разработки информационно-расчетных задач на основе специализированной модели.
3. Предложить адекватные способы представления и исследования иерархических структур данных в рамках реляционной модели на основе алгебраических спецификаций и функционального языка программирования .
4. Разработать средства, обеспечивающие адекватную реализацию модели в среде систем JAVA, Visual FoxPro и MS SQL .
5. Построить демонстрационную модель на примере системы автоматизации учета автомобильного транспорта с целью обоснования работоспособности предлагаемых в диссертации средств и методов.

Методы исследования

В работе использованы методы реляционной алгебры, математической логики, элементы системного анализа, прикладного программирования, а также современные методологии организации иерархических систем.

Модель построена на основе сочетания средств реляционной алгебры с аппаратом алгебраических спецификаций и средств функционального программирования. Язык спецификаций модели обеспечивает формальное описание внутренней структуры классов объектов, включающей описание атрибутов и операций, а также поведения соответствующих процессов

Научная новизна

1. Создание специализированной модели для класса информационно-расчетных задач на основе расширения реляционной модели средствами алгебраических спецификаций и функционального языка программирования

2. Создание интегрированной среды разработки информационно-расчетных задач на основе специализированной модели.

3. Разработка адекватных способов представления и исследования иерархических структур данных в рамках реляционной модели на основе алгебраических спецификаций и функционального языка программирования.

4. Разработка средств, обеспечивающих адекватную реализацию модели в среде систем JAVA, Visual FoxPro и MS SQL .

Практическая ценность результатов

На защиту выносятся

1. Специализированная модель для класса информационно-расчетных задач на основе расширения реляционной модели средствами алгебраических спецификаций и функционального языка программирования

2. Интегрированная среда разработки информационно-расчетных задач на основе специализированной модели..

3. Разработанные средства, обеспечивающие адекватную реализацию

модели в среде систем JAVA, Visual FoxPro и MS SQL .

4. Демонстрационная модель на примере системы автоматизации учета автомобильного транспорта с целью обоснования работоспособности предлагаемых в диссертации средств и методов

Апробация работы

Основные научные результаты диссертационной работы были доложены на научных семинарах кафедры теоретической кибернетики и на XV Международной конференции Проблемы теоретической кибернетики (Казань, Россия, 2–7 июня, 2008).

Публикации

По теме диссертации опубликовано 5 работ, в том числе 1 – в журнале, входящем в Перечень ВАК РФ.

Структура и объем работы

Диссертация состоит из введения, четырех глав, заключения и списка использованной литературы из 87 наименований, включая работы автора. Объем диссертации составляет 118 страниц машинописного текста.

Краткое содержание работы

Введение обосновывает актуальность темы, излагаются цели, задачи исследования, новизна и практическая ценность выносимых на защиту результатов.

В первой главе автором производится анализ литературы по сходной тематике для обоснования применения представляемых в диссертационной работе формализованных средств для построения модели. Также автором анализируются различные способы представления моделей (неформальные, полуформальные и формальные), их преимущества и недостатки. В качестве основного вывода предложено заключение о необходимости расширения аппарата реляционной алгебры посредством включения в него средств функционального программирования и алгебраических спецификаций с целью построения адекватной модели для класса информационно-расчетных задач.

Во второй главе предлагаются средства для построения моделей иерархических структур данных на основе алгебраических спецификаций и средств функционального языка программирования с целью последующей реализации модели в реляционной среде. Данные средства предлагаются в связи с тем, что в процессах реализации информационно-расчетных задач традиционно используется реляционная модель представления данных. Однако в этих задачах часто возникает необходимость использования иерархических структур. Для обеспечения преемственности реляционных моделей возникает необходимость их расширения путем включения адекватных средств представления и обработки иерархических структур.

Спецификация обеспечивает формальное описание внутренней структуры классов объектов, включающей описание атрибутов и операций.

Для описания структуры данных и операций использованы средства функционального языка программирования на основе языка LISP.

Любую таблицу базы данных можно представить в виде списка (S,I,T), где S- объект, определяющий структуру таблицы, I –список индексных выражений, а T – непосредственно сама таблица. Объекты S,I и T определяются следующим образом:

$S=(s_1, s_2, \ldots, s_m)$, где s_i (i=1..m) – описания атрибутов таблицы. Каждое описание атрибута представим в виде списка $s=(a,c, l_1, l_2)$, где a – наименование атрибута, c - тип атрибута, l_1 , l_2 – модификаторы длины. Например, (NAME,Char,50,0) – означает атрибут с именем NAME символьного типа с максимальной длиной в 50 символов, (SALARY,Numeric,12,2) - атрибут с именем SALARY численного типа с максимальной длиной в 12 позиций, включая десятичную точку, и 2 –цифры после десятичной точки.

$I=(i_1, i_2, \ldots, i_k)$, где i_j (j=1..k)- индексное выражение, представляемое в виде выражения $E(a_1, a_2, \ldots, a_l)$, где a_i (i=1..l) – атрибуты таблицы.

$T=(t_1, t_2, \ldots , t_n)$ Через nil будем обозначать пустую таблицу. Пусть t – запись таблицы T, то есть $t \in T$, тогда любую запись t можно представить в виде последовательности $t =(a_1,a_2, \ldots , a_k)$, где $a_1,a_2, \ldots , a_k$ – атрибуты.

Определим специальную функцию struct(T), значением которой будет являться структура таблицы T, а через index(T) - список индексных выражений таблицы T. Далее, для простоты изложения последующего материала будем в качестве таблицы рассматривать только объект T, имея в виду, что для любого такого объекта мы можем всегда определить его структуру посредством функции struct(T), а список индексных выражений - через функцию index(T) . Структурную и индексную части будем рассматривать только в случае необходимости.

Определим следующие операции:

1. seta(t,i,v) – присвоение значения v i-ому атрибуту записи t
seta(t,i,v) $=_{df}$ IF(i=1,cons(v,tail(t),seta(tail(t),i-1,v))), причем
i=1..n, где n- число атрибутов записи t;
2. new(S) – значением функции является пустая запись, построенная в соответствии со структурой S;
3. create(S,I) – создание таблицы со структурой S и списком индексов I, то естьT= create(S,I) ⇔ struct(T)=S & index(T)=I ;
4. appendb(T) – добавление новой пустой записи к таблице T,
appendb(T) $=_{df}$ cons(new(struct(T)),T);
filter(T,b) – значением функции является новая таблица, включающая в себя все те записи из таблицы T, которые удовлетворяют условию b,
5. filter(T,b) $=_{df}$ IF(cond(Head(T),b),cons(Head(T),filter(tail(T),b), filter(tail(T),b)) ,
где cond(t,b) - функция, принимающая булевское значение в результате вычисления логического выражения b для записи t;
6. delete(T) – значением функции является новая таблица, которая получается в результате удаления головной записи из таблицы T, то есть
delete(T) $=_{df}$ tail(T)

Следующая ниже алгебраическая спецификация описывает структуру таблицы:

Object Table:
$Table = nil \mid (List\ of\ record)$
$List\ of\ record = record \mid record, List\ of\ record$
$record = (List\ of\ Aribute)$
$List\ of\ Aribute = Aribute \mid Aribute, List\ of\ Aribute$
$Aribute = Ident$
$Ident = Letter \mid IdentLetter \mid IdentDigit$
Functions
$seta(Record, natural, Atom) \to record$
$new(structure) \to record$
$create(structure, index) \to Table$:
$structure = List\ of\ def$
$def = (Name, Type, Len1, Len2)$
$Name = Ident$
$Type = Ident$
$Len1 = natural$
$Len2 = natural$
$appendb(Table) \to Table$
$delete(Table) \to Table$
$filter(Table, boolean) \to Table$

Axioms
$create(S, I) =_{df} T = create(S, I) \Leftrightarrow struct(T) = S \,\&\, index(T) = I$
$appendb(T) =_{df} cons(new(struct(T)), T)$
$filter(T, b) =_{df} IF(cond(Head(T), b), cons(Head(T), filter(tail(T), b), filter(tail(T), b)))$
$delete(T) =_{df} tail(T)$
END

Далее предлагаются способы представления иерархических структур на основе приведенных выше формализованных средств.

Традиционные способы представления иерархических структур в реляционных моделях сводятся к следующим: иерархическая структура в рамках одной таблицы, иерархическая организация которой определяется специальным способом кодирования ключевого атрибута таблицы, и иерархическая структура на основе отношений между двумя и более таблицами.

Ниже приводится описание 3-х способов представления иерархических структур, из которых первые 2 способа основываются на представлении

иерархии в рамках одной таблицы, а последний - на представлении иерархии в рамках двух и более таблиц.

Описание модели иерархических структур приводится на основе определенных выше формализмов.

Первый способ представления иерархии

При этом способе представления ключевое выражение таблицы, однозначно идентифицирующее каждую запись таблицы, определяется парой атрибутов

<Id, Par>, где Id – уникальный номер записи, а Par – номер родительской записи. Пусть T –некоторая иерархическая таблица, представленная в виде последовательности записей $T=(t_1, t_2, \dots, t_n)$. Через nil будем обозначать пустую таблицу. Пусть t – запись таблицы T, то есть $t \in T$, тогда любую запись t можно представить в виде последовательности $t =(Id,Par, a_1,a_2, \dots, a_k)$, где $a_1,a_2, \dots, a_k$ – все остальные атрибуты. Определим функции fid(t) для выделения собственного номера записи t и fpar(t) для выделения номера родительской записи, то есть fid(t)= Id и fpar(t)=Par. Запись, определяющая начальный уровень иерархии не имеет родителя, поэтому в качестве номера родительской записи будет 0. Для описания модели были использованы средства языка функционального программирования.

Для данного способа определены основные операции для иерархических структур.

Это – следующие операции:

1. maxcode(T) – максимальное значение номеров записей таблицы T
2. add(T,t,p) – добавление записи t в конец таблицы T с формированием
   a. в записи t собственного номера и номера родительской записи p
3. del(T,k) – удаление из таблицы T записи с идентификационным номером k, при этом вместе с упомянутым элементом удаляются все связанные записи с более низкими уровнями иерархии (дочерние, дочерние дочерних и так до самого конца). Такой способ удаления обеспечивает сохранение целостности иерархической структуры, которая предусматривает наличие родителя для каждого элемента, кроме начального. Начальный элемент является единственным, у которого вместо номера родительского элемента указывается нулевое значение.
4. down(T,k) – обеспечивается выделение всех дочерних записей по отношению к записи с идентификационным номером k
5. up(T,k) – обеспечивается возврат на предыдущий (родительский) уровень иерархии относительно дочерней записи с идентификационным номером k посредством выделения записей, принадлежащих одному и тому же родителю

Следующая ниже алгебраическая спецификация полностью описывает

структуру объектов и основные функции первого способа представления.

Object Iertable :

$Iertable = nil \mid (List\ of\ record)$

$List\ of\ record = Record \mid Record, List\ of\ record$

$record = (Seq)$

$Seq = Id, Par \mid Seq, SeqA$

$Id = natural$

$Par = natural$

$SeqA = Aribute \mid SeqA, Aribute$

$Aribute = Ident$

$Ident = Letter \mid IdentLetter \mid IdentDigit$

Functions

$fid(Record) \to natural$

$fpar(Record) \to natural$

$\max code(Iertable) \to natural$

$add(Iertable, Record, natural) \to Iertable$

$del(Iertable, natural) \to Iertable$

$down(Iertable, natural) \to Iertable$

$up(Iertable, natural) \to Iertable$

Axioms

$t = (Id, Par, a_1, a_2, ..., a_k) \Rightarrow fid(t) = Id\ \&\ fpar(t) = par$

$T \neq nil \Rightarrow maxcode(T) =_{df} m = maxcode(T) \Leftrightarrow$

$(\exists t \in T(m = fid(t))) \& (\forall w \in T \Rightarrow m \geq fid(w))$

$maxcode(nil) = 0$

$add(T, t, p) =_{df} T = (t_1, t_2, ..., t_n) \Rightarrow add(T, t, p) = (t_1, t_2, ..., t_n, fill(T, t, p))$

$fill(T, t, p) =_{df} t = (Id, Par, a_1, a_2, ..., a_k) \Rightarrow$

$fill(T, t, p) = (\max code(T) + 1, p, a_1, a_{2,}, ..., a_k)$

$del(T, k) =_{df}$ IF(T = nil, nil, IF(fid(head(T)) = k OR Isparg(T, head(T), k)),

del(tail(T), k), cons(head(T), del(tail(T), k))))

$\mathrm{Isp\,arg}(\mathrm{T},\mathrm{t},\mathrm{k}) =_{\mathrm{df}} (\mathrm{t} \in T)\ \&\ (Par(t) = k\ \mathrm{OR}\ (\exists w \in T\ (\mathit{fid}(w) = \mathit{fpar}(t)\ \&$

$\mathit{Isp}\,\mathrm{arg}(T, w, k)))$

$down(T, \kappa) =_{df} IF(T = nil, nil, IF(\ \mathit{fp}\,\mathrm{arg}(head(T)) \neq k), del(tail(T), k),$

$cons(head(T), del(tail(T), k))))$

$up(T, k) =_{df} IF(T = nil, nil, IF(\ \mathit{fpar}(head(T)) = nParPar(T, k),$

$cons(head(T), up(tail(T), k)), up(tail(T), k)))$

$nPar(T, k) =_{df} IF\ (T = nil, 0, IF(\ \mathit{fid}(head(T)) = k,$

$\mathit{fpar}(head(T)), nPar(tail(T), k)))$

$nParPar(T, k) =_{df} nPar(T, nPar(T, k))$

END

Второй способ представления иерархии

При этом способе представления ключевое выражение таблицы, однозначно идентифицирующее каждую запись таблицы, определяется атрибутом Id, где Id=(n_1 ,n_2 ,n_3 ,….,n_k) – где n_i (i=1..k)– натуральные числа, определяющие номера вершин на каждом уровне иерархии. Таким образом данный атрибут представляет из себя список, однозначно определяющий полный путь соответствующей записи, начиная от корня иерархии до текущей записи. Поскольку идентификационное выражение для каждой записи представляется в виде списка, то вместо понятия идентификационного выражения будем употреблять термин «идентификационный список», или для сокращения – «идентификатор». Последнее число идентификатора записи (n_k) будем называть собственным номером записи. Для данного способа также определены те же самые основные операции над иерархическими структурами, что и в первом способе. Однако определения и реализация этих операций являются другими в соответствии с учетом своей специфики кодировки.

Следующая ниже алгебраическая спецификация полностью описывает структуру объектов и основные функции второго способа представления.

Object Iertable:

$\text{Iertable} = \text{nil} \mid (\text{List of record})$

$\text{List of record} = \text{Record} \mid \text{Record, List of record}$

$\text{Record} = (\text{Seq})$

$\text{Seq} = \text{Id} \mid \text{Seq}, \text{SeqA}$

$\text{Id} = (\text{List of natural}) \mid \text{nil}$

$\text{List of natural} = \text{natural} \mid \text{natural, List of natural}$

$\text{SeqA} = \text{Attribute} \mid \text{Attribute,SeqA}$

$SeqA = Aribute \mid SeqA, Aribute$

$Aribute = Ident$

$Ident = Letter \mid IdentLetter \mid IdentDigit$

Functions

$\text{fid}(\text{Record}) \to \text{Id}$

$\text{fpar}(\text{Record}) \to \text{Id}$

$\text{maxcode}(\text{Iertable, Id}) \to \text{natural}$

$\text{add}(\text{Iertable, Record, Id}) \to \text{Iertable}$

$\text{del}(\text{Iertable, Id}) \to \text{Iertable}$

$\text{down}(\text{Iertable, Id}) \to \text{Iertable}$

$\text{up}(\text{Iertable, Id}) \to \text{Iertable}$

Axioms

$\text{fid}(t) =_{df} t = (\text{Id}, a_1, a_2, \ldots, a_k) \Rightarrow \text{fid}(t) = \text{Id}$

$\text{fpar}(t) =_{df} D(\text{fid}(t))$

$D(s) =_{df} \text{If } (\text{tail}(s) = \text{nil}, \text{nil}, \text{cons}(\text{head}(s), D(\text{tail}(s))))$

$T \neq \text{nil} \Rightarrow \text{maxcode}(T,s) =_{df} (m = \text{maxcode}(T,s)) \Leftrightarrow$

$(\exists t \in T(m = last(fid(w)) \;\&\; fpar(t) = s) \;\&\; (\forall w \in T \;\&\; fpar(w) = s) \Rightarrow$

$m \geq last(fid(w)))$

$last(S) =_{df} IF(tail(S) = nil, head(S), last(tail(S))),$

$\text{maxcode}(nil, s) = 0$

$add(T,t,p) =_{df} T = (t_1, t_2, \ldots, t_n) \Rightarrow add(T,t,p) = (t_1, t_2, \ldots, t_n, fill(T,t,p))$

$\text{fill}(T,t,p) =_{df} t = (\text{Id}, a_1, a_2, \ldots, a_k) \Rightarrow$

$\text{fill}(T,t,p) = (\text{append}(\text{fid}(p), (\text{maxcode}(T,p)+1)), a_1, a_2, \ldots, a_k)$

$\text{del}(T,s) =_{df} \text{IF}(T = \text{nil}, \text{nil},$

$\text{IF}(\text{prefix}(\text{fid}(\text{head}(T)), s), \text{del}(\text{tail}(T), s), \text{cons}(\text{head}(T), \text{del}(\text{tail}(T), s))))$

$\text{prefix}(s,p) =_{df} \text{IF}(s = \text{nil}, \text{true}, \text{IF}(\text{head}(s) = \text{head}(p), \text{prefix}(\text{tail}(s), \text{tail}(p)), \text{false})))$

$\text{down}(T,s) =_{df} \text{IF}(T = \text{nil}, \text{nil}, \text{IF}(\text{fpar}(\text{head}(T)) = s, \text{cons}(\text{head}(T), \text{tail}(T)),$

$\text{down}(\text{tail}(T), s)))$

$up(T,s) =_{df}$ IF(T = nil, nil, IF(fpar(head(T)) = nParPar(T,s), cons(head(T), up(tail(T),s)), up(tail(T),s)))

$nPar(T,s) =_{df}$ IF (T = nil, nil, IF(fid(head(T)) = s, fpar(head(T))), nPar(tail(T),s)))

nParPar(T,s) = df nPar(T, nPar(T,s))

END

2.3 Третий способ.

Данный способ представления иерархии основывается на создании специальных отношений между 2-мя или более таблиц. Здесь ограничимся рассмотрением 2-х таблиц, имея в виду, что все результаты, получаемые в этом случае легко можно обобщить для большего количества таблиц. Такие отношения между таблицами классически делятся на виды:

«один ко многим», «один к одному» и «многие ко многим». Для представления иерархии будем рассматривать только первый из перечисленных выше видов отношений. При этом способе представления ключевое выражение первой таблицы, определяется индексным выражением, однозначно идентифицирующее каждую запись таблицы, а в качестве 2-ого элемента отношения выбирается соответствующее индексное выражение второй таблицы. Наложим следующие ограничения на структуры упомянутых выше таблиц:

1)список атрибутов первой из таблиц T_1 будем представлять в виде

(Id, a_1,a_2, ... , a_k), где Id- ключевой атрибут, a_1,a_2, ... , a_k – все остальные атрибуты таблицы T_1; определим функцию fid(t) для выделения уникального номера записи t, то есть fid(t)= Id, причем $t \in T_1 \& s \in T_1 \& s \neq t \Rightarrow fid(t) \neq fid(s)$;

2) список атрибутов второй из таблиц T_2 будем представлять в виде

(Par, a_1,a_2, ... , a_m), где Par – номер родительской записи в таблице T_1, a_1,a_2, ... , a_m – все остальные атрибуты таблицы T_1 ; функция fpar(t) определяет номер родительской записи в таблице T_1 причем $s \in T_2 \Rightarrow (\exists t \in T_1).(fpar(s)= fid(t))$ (нарушение этого условия рассматривается как нарушение целостности соответствующей базы данных).

Таким образом отношение иерархии между таблицами T_1 и T_2 можно определить следующим образом:

$$H(T_1, T_2) =_{df} (t \in T_1 \& s \in T_1 \& s \neq t) \Rightarrow fid(t) \neq fid(s)) \&$$
$$(s \in T_2 \Rightarrow (\exists t \in T_1).(fpar(s) = fid(t)))$$

Следующий ниже пример демонстрирует отношение иерархии между двумя таблицами (отношение вида «один ко многим»):

Таблица T_1(список университетов)

Id Name

1	Казанский университет		
2	Московский университет		

Таблица T_2 (список факультетов)

Par Name

1	ВМК	
1	ФИЗФАК	
2	ВМК	
2	ФИЗФАК	

В последующем таблицу T_1 будем называть родительской, а таблицу T_2-дочерней.

Определим основные операции для иерархических структур.

1.Add1(T,t) – добавление записи t в конец таблицы T с формированием в записи t уникального значения ключевого атрибута, то есть

$$T = (t_1, t_2, ..., t_n) \Rightarrow add1(T, t, p) = (t_1, t_2, ..., t_n, fill1(T, t)), где если$$
$$t = (Id, a_1, a_2, ..., a_k), то fill1(T, t) = (\max code(T) + 1, a_1, a_2, ..., a_k).$$

maxcode(T) – максимальное значение номеров записей таблицы T,

то есть:

$$T \neq nil \Rightarrow \max code(T) =_{df} (m = \max code(T)) \Leftrightarrow$$
$$(\exists t \in T(m = fid(t))) \& (\forall w \in T \Rightarrow m \geq fid(w)),$$
$$\max code(nil) = 0$$

Эта операция применима только к таблице типа T_1.

2.Add2(T_1,T_2,t,p) – добавление записи t в конец таблицы T_2 с формированием в записи t номера родительской записи p, то есть

T_2= (t_1, t_2,…, t_n) & ($\exists t \in T_1$)(fid(t)= p) =>

Add2(T_1,T_2,t,p) = (t_1, t_2,…, t_n,fill2(t,p)) ,

где fill2(t,p)=(p , a_1,a_2, ... , a_k)

3. del1(T,k) – удаление из таблицы T записи с номером k, то есть

$$del1(T,k) =_{df} IF(T = nil, nil, IF(\ fid(head(T)) = k\ , del1(tail(T),k), cons(head(T), del1(tail(T),k))))$$

Эта операция применима только к таблице типа T_1.

4.Delcs(T_1,T_2,k) – удаление из таблицы T_1 записи с номером k. При удалении записи из родительской таблицы в дочерней таблице происходит удаление всех соответствующих дочерних записей.

Таким образом:

Delcs(T_1,T_2,k) =$_{df}$ (T_1'=Del1(T_1,k) & T_2'= Del2(T_2,k)), где T_1' и T_2' определяют новые состояния таблиц T_1 и T_2 после выполнения операции удаления,

Del2(T,k)) =$_{df}$ IF(T=nil, nil, IF(fpar(head(T))=k, Del2(tail(T),k), cons(head(T), Del1(tail(T),k)).

Функция Delcs(T_1,T_2,k) определяет так называемое каскадное удаление (CASCADE). Выполнение этой операции возможно лишь в том случае, если на соответствующее отношение не наложено ограничение на удаление (RESTRICT). В противном случае из родительской таблицы разрешено удалять только те записи, для которых нет дочерних записей.

5.Down1(T_1,T_2,k) – обеспечивается выделение всех тех записей из таблицы T_2 , являющихся дочерними по отношению к записи таблицы T_1 с номером k. То есть

Down1(T_1,T_2,k) =$_{df}$ IF(IsPar1(T_1,k), Down2(T_2,k), nil),

IsPar1(T,k) =$_{df}$ ($\exists t \in$ T).(fid(t)=k) (функция проверки существования записи с номером k в таблице T)

Down2(T,k) =$_{df}$ IF(T=nil, nil, IF(fpar(head(T))=k, cons(head(T),

Down2 (tail(T),k)) , Down2 (tail(T),k)))

6.Up1(T_1,T_2) обеспечивается возврат на родительский уровень иерархии, Up1(T_1,T_2) =$_{df}$ T_1 .

Object HREL:

$HREL = H(Table1, Table2)$

$Table1 = nil|(List\ of\ record1)$

$List\ of\ record1 = record1 \ | record1,\ List\ of\ record1$

$record1 = (Seq1)$

$Seq1 = Id|Seq1,SeqA$

$Table2 = nil|\ (List\ of\ record2)$

$List\ of\ record2 = record2|record2,List\ of\ record2$

$record2 = (Seq2)$

$Seq2 = Par|Seq2,SeqA$

$Id = natural$

$Id = natural$

$Par = natural$

$SeqA = Aribute|SeqA,Aribute$

$Aribute = Ident$

$Ident = Letter|Ident\ Letter \ |Ident\ Digit$

Functions

$H(Table1, Table2) \rightarrow boolean$

$Add1(Table1, record1) \rightarrow Table1$

$fill1(Table1,record1) \rightarrow Table1$

$fill2(Table2,record2) \rightarrow Table2$

$Add2(Table1, Table2,record2,Natural) \rightarrow Table2$

$\text{maxcode}(Table1, \text{Id}) \rightarrow \text{natural}$

$del1(Table1, natural) \rightarrow Table1$

$Del2(Table2,natural) \rightarrow Table2$

$Delcs(Table1 ,natural) \rightarrow boolean$

$Down1(Table1,Table2,natural) \rightarrow Table2$

$Down2(Table2,natural) \rightarrow Table2$

$Up1(Table1, Table2) \rightarrow Table1$

Axioms

$add1(T, t, p) =_{df}$

$T = (t_1, t_2, ..., t_n) \Rightarrow add1(T, t, p) = (t_1, t_2, ..., t_n, fill1(T, t))$

$fill1(T, t) =_{df}$

$t = (Id, a_1, a_2, ..., a_k) \Rightarrow fill1(T, t) = (\max code(T) + 1, a_1, a_2, ..., a_k).$

$T \neq nil \Rightarrow \max code(T) =_{df} (m = \max code(T)) \Leftrightarrow$

$(\exists t \in T(m = fid(t))) \& (\forall w \in T \Rightarrow m \geq fid(w)),$

$\max code(nil) = 0$

$Add2(T1, T2, t, p) =_{df}$

$T2 = (t1, t2, \ldots, tn) \& (\exists t \in T1)(fid(t) = p) \Rightarrow$

$Add2(T1, T2, t, p) = (t1, t2, \ldots, tn, fill2(t, p))$

$fill2(t, p)) =_{df}$

$t = (Par, a_1, a_2, ..., a_k) \Rightarrow fill2(t, p)) = (p, a1, a2, \ldots, ak)$

$del1(T, k) =_{df} IF(T = nil, nil, IF(fid(head(T)) = k, del1(tail(T), k),$

$cons(head(T), del1(tail(T), k))))$

$Del2(T,k) =_{df} IF(T = nil, nil, IF(fpar(head(T)) = k, Del2(tail(T), k),$

$cons(head(T), Del1(tail(T), k))$

$Delcs(T1, T2, k) =_{df} (T1' = Del1(T1, k) \& T2' = Del2(T2, k))$

$Down1(T1, T2, k) =_{df} IF(IsPar1(T1, k), Down2(T2, k), nil)$

$IsPar1(T, k) =_{df} (\exists t \in T).(fid(t) = k)$

$Down2(T, k) =_{df} IF(T = nil, nil, IF(fpar(head(T)) = k, cons(head(T),$

$Down2(tail(T), k)), Down2(tail(T), k)))$

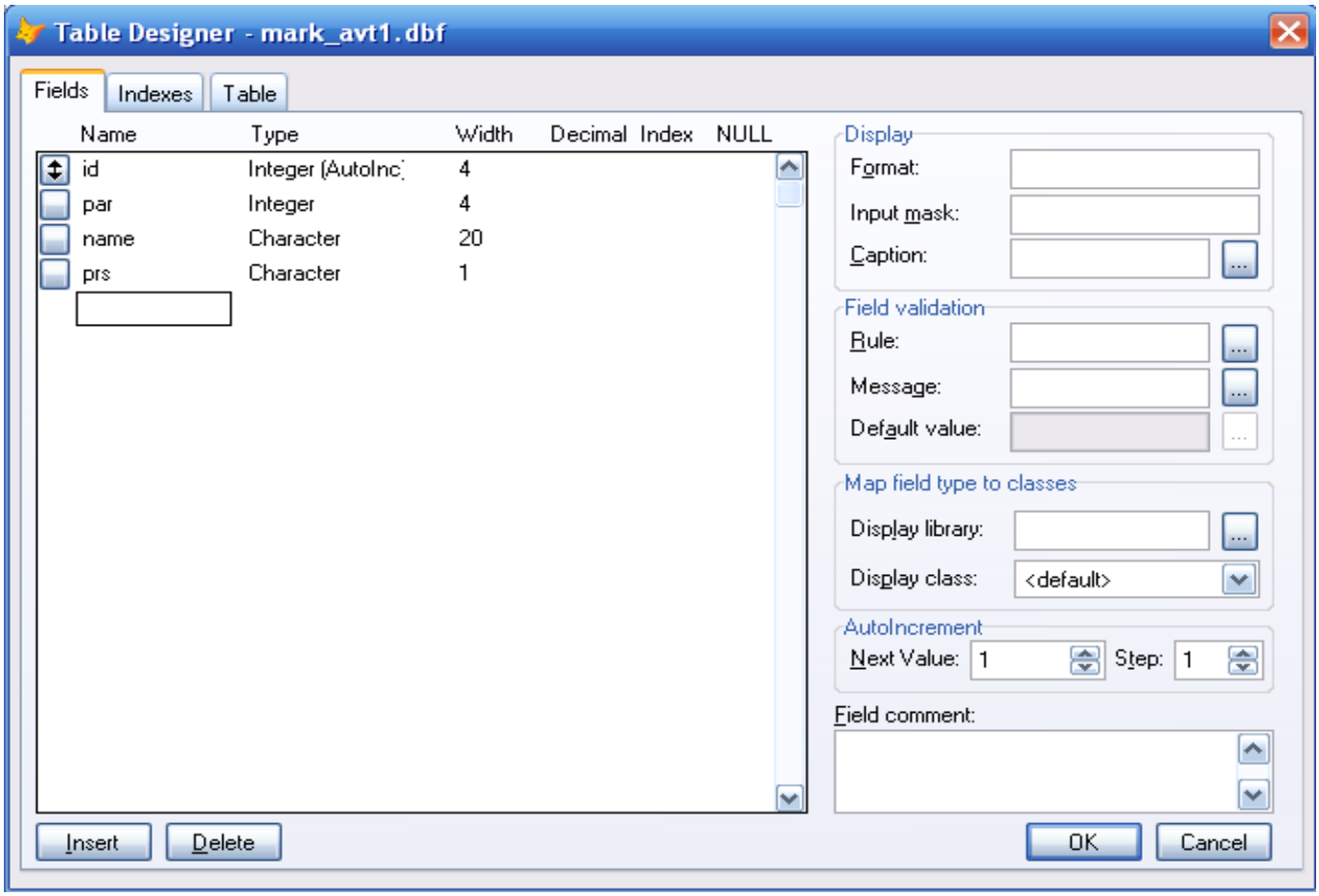
$Up1(T1, T2) =_{df} T1.$

END

Использование приведенного выше способа представления иерархии для больших глубин иерархии является нецелесообразным, так как требует привлечения большого количества таблиц и соответственно может серьезно снизить эффективность обработки иерархических структур.

Третья глава описывает разработанные автором средства, обеспечивающие перевод формализованной модели в соответствующие средства СУБД Visual Foxpro, MS SQL и JAVA среды. Кроме этого предлагаются средства преобразования данных таблицы представления иерархии второго типа в данные таблицы представления иерархии первого типа.

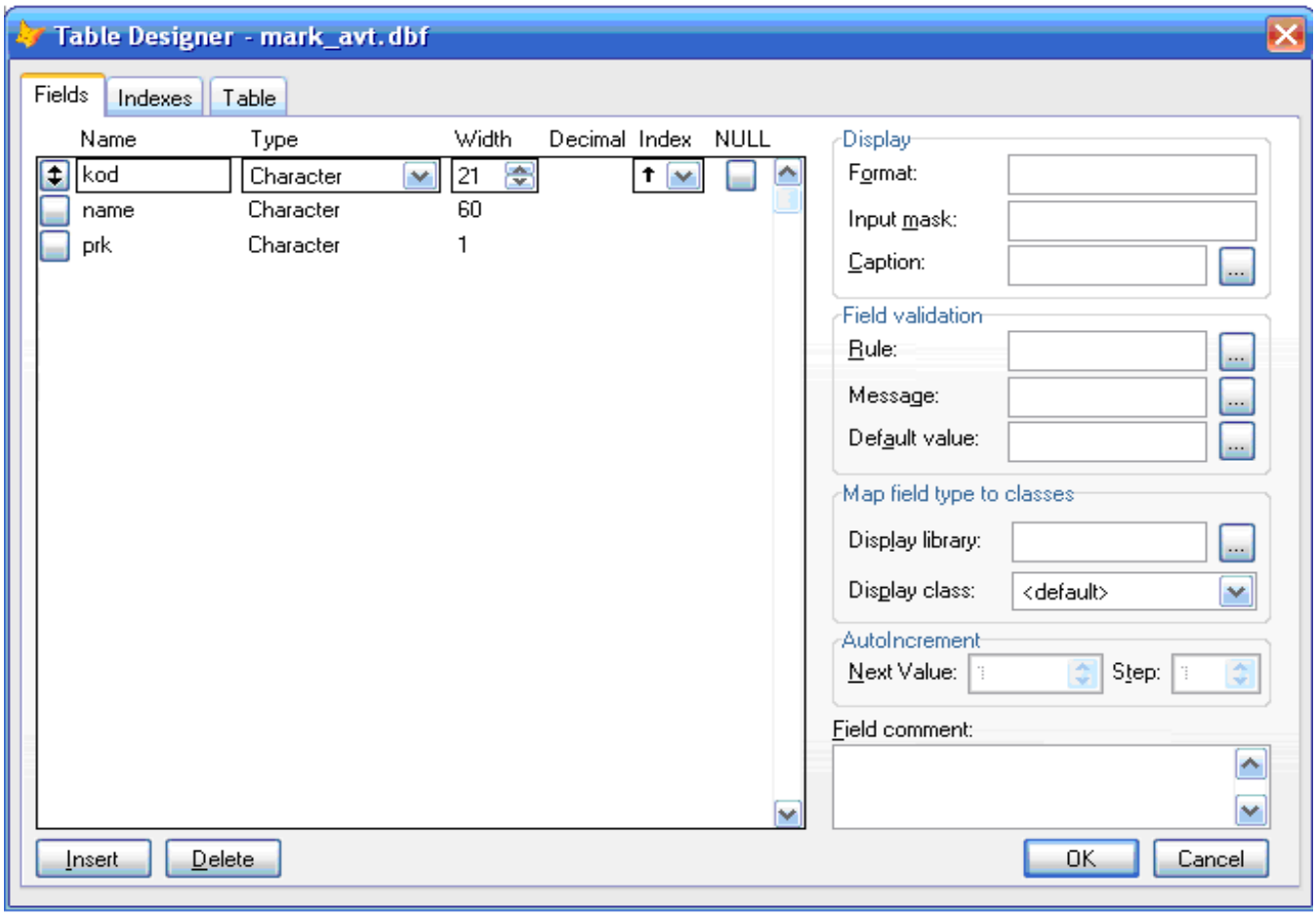
Структура таблицы для первого способа представления иерархии:



Для данной таблицы:

- Id – уникальный номер записи, имеет числовое значение
- Par – номер родительской записи, имеет числовое значение
- Name – имя записи, имеет символьное значение
- Prs – признак записи, указывающий наличие дочерних элементов, имеет символьное значение

Структура таблицы для второго способа представления иерархии:

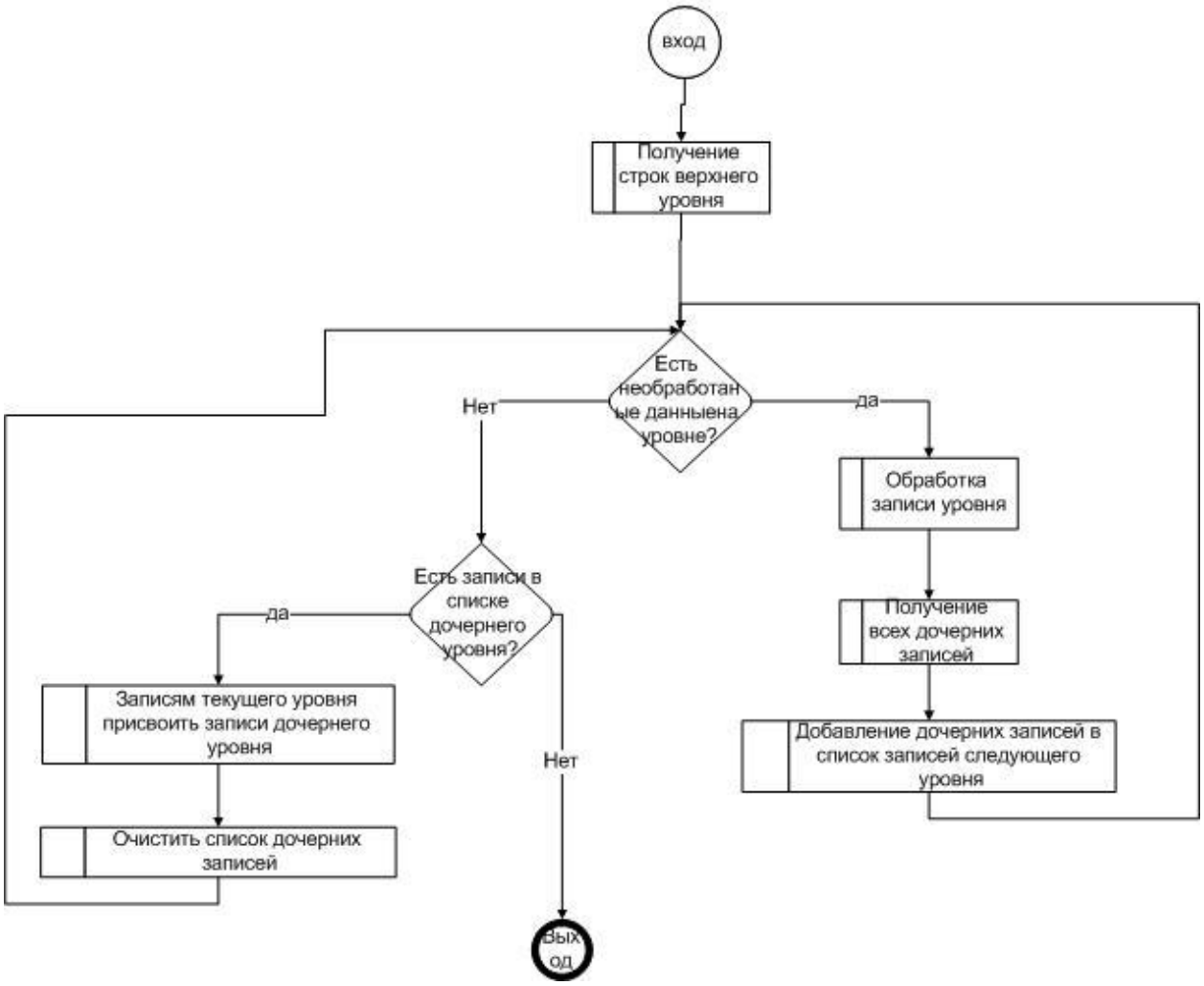


Для данной таблицы:

- kod – уникальный номер записи, имеет символьное значение
- Name – имя записи, имеет символьное значение
- Prs – признак записи, указывающий наличие дочерних элементов, имеет символьное значение

Модуль преобразования данных таблицы представления иерархии второго типа в данные таблицы представления иерархии первого типа.

Схематично процесс преобразования данных можно изобразить в виде следующей блок-схемы.



Данный модуль использует методы, ранее реализованные в классах представления таблиц, для получения данных содержащихся на различных уровнях таблиц.

Для каждого уровня иерархии таблицы второго типа происходит считывание всех записей данного уровня и сопоставление их идентификаторов, соответствующих идентификаторам в таблице первого типа.

В четвертой главе приводится описание демонстрационной модели на примере системы автоматизации учета автомобильного транспорта с целью обоснования работоспособности предлагаемых в диссертации средств и методов.

Заключение

В соответствии с поставленной задачей в диссертационной работе создана специализированная модель для класса информационно-расчетных задач на основе расширения реляционной модели средствами алгебраических спецификаций и функционального языка программирования. Также создана интегрированная среда разработки информационно-расчетных задач на основе построенной специализированной модели и предложены адекватные способы представления и исследования иерархических структур данных в рамках реляционной модели на основе алгебраических спецификаций и функционального языка программирования.

Основная особенность упомянутой интегрированной среды разработки заключается в том, что она включает в себя наряду с формализованными средствами модели средства реализации модели в программную среду СУБД Visual Foxpro, MS SQL и системы JAVA.

Благодарности. Автор выражает искреннюю благодарность своему научному руководителю Арслану Ильясовичу Еникееву за постоянное внимание и неизменную поддержку данной работы.

Публикации по теме диссертации

Работа опубликованная в журнале, входящем в Перечень ВАК :

1. **Камашев М.А. Специализированные модели для разработки программных приложений на основе алгебраических спецификаций и средств функционального программирования// Системы управления и информационные технологии. Научно-технический журнал № 4(42). – Воронеж: Изд-во «Научная книга», 2010. – С. 73-78 ISSN 1729-5068.**

Другие публикации :

2. А. И. Еникеев, Т. Бендума, М. А. Камашев. Специализированные объектно-ориентированные модели программных систем. // Материалы XV Международной конференции Проблемы теоретической кибернетики (Казань, Россия, 2–7 июня, 2008). – Казань: Изд-во «Отечество», Июнь 2008. – С. 36-37.
3. Камашев М.А, Специализированная обьектно-ориентированная модель для представления иерархических структур// Информационные технологии моделирования и управления. Международный научно-технический журнал № 5(64). –Воронеж: Изд-во «Научная книга», 2010. - С. 669-678. ISSN 1813-9744
4. Камашев М.А. // О технологии создания специализированных объектно-ориентированных приложений Исследования по информатике. Вып. 11. – Казань: Изд-во “Отечество”, 2007. - С. 123-128.

5. Камашев М.А. // Описание программного обеспечения// Техника и технология", №4 (июль)– Москва: Изд-во “Спутник+”, 2011. - С. 17-38